Tableau Coursework Brief

COMP0009: Logic

Deadline: 29 November 2024, 23:59 UK Time

1 Technical Brief

For this assignment, you will be asked to implement a propositional and first order logic tableau, using Python. Your program will be run on an unseen input file and tested against its corresponding model output. Your program will be expected to correctly identify the type of a propositional/first order logic formula, and identify whether it is satisfiable, not satisfiable, or (in the case of FOL tableau) satisfiablility cannot be determined after introducing 10 new constants in the δ-expansions on any given open branch.

IMPORTANT:

- Your file must not include any import statements. If your submission contains an import statement, it will be considered as cheating, you will receive a mark of 0.
- All submissions must be submitted as a single plain ASCII (CP 437) encoded file names tableau.py that uses either CRLF or LF line endings. Failure to adhere to the submission format requirements will additionally result in your receipt of a 0 mark. All of these settings can be easily set in Visual Studio Code and we suggest you set them there.

Below we define both the language of propositional and first order logic. We limit our propositional letters to p, q, r, s, and our Binary connectives to conjunctions, disjunctions, and implications. White space or extra brackets are not allowed in formulas.

```
FMLA := PROP                    (Proposition)
      | ~FMLA                   (Negation)
      | (FMLA*FMLA)             (Binary Connective)

PROP := p|q|r|s    * := /\|\/|=>   (and, or, implies)
```

Similarly for FOL, we limit our variables to x, y, z, w, no function symbols, and the only predicates are binary predicates P, Q, R, S.

```
var := x|y|z|w
PRED := P|Q|R|S      * := /\|\/|=>
FMLA := PRED(var,var) (Atom)
      | ~FMLA         (Negation)
      | EvarFMLA      (Existentially Quantified)
      | AvarFMLA      (Universally Quantified)
      | (FMLA*FMLA)   (Binary Connective)
```

To see some examples, see sample input/output files. The input file should have the first line containing PARSE in order for the program to produce parser output, or SAT to produce satisfiability output (or both). The remaining lines should be propositional or first order formulas.

If a formula is parsed as a formula, you should construct a Tableau and see if it has an open branch. For propositional formulas, you will be asked to determine whether or not it is satisfiable. For first order logic formulas, you may assume there are no free variables. You will be asked to try and determine satisfiability, however, the tableau might never close. Thus, if you are required to add more than 10 new constants on all open branches under some reasonable fair schedule, you may leave satisfiability undetermined.

2 Assessment Criteria

You must submit a single file, tableau.py. It will be tested against five sample input files - it should be able to run as is on the lab machines using the below command, provided there is an input.txt file in the same directory.

```
python3 tableau.py > output.txt
```

A test will pass if and only if the output.txt file matches the model output file. If your program does not finish computing within a minute, it will timeout and the test will fail. You will be awarded marks according to the table below:

Task	**Marks**
Correctly parse propositional formulas	20
Correctly determine SAT for propositional formulas	20
Correctly parse first order logic formulas	20
Correctly determine SAT for FOL sentences that will not result in a loop under any schedule.	20
Correctly determine SAT for any FOL sentence, up to assigning 10 variables	20

TESTING YOUR SUBMISSION Before submitting your code, you will be able to test it via a web app as many times as you'd like. It will show your output.txt file for the public test cases as well as the mark your submission would get when tested against the five secret input files.

Note that we monitor these submissions and that we will investigate any suspicions we may have of cheating by using this app in ways other than it was intended to be used. Anyone found cheating will receive a mark of 0 and may be referred to the academic misconduct board.

Make sure to test your submission from the cs network. You may choose to use the cs vpn services as described in the CSWall section of `https://tsg.cs.ucl.ac.uk/working-from-home/`. Once connected, you will be able to use the app at `http://logic2023.cs.ucl.ac.uk/`.